The Role of Trust in Information Integrity Protocols*

G. J. Simmons
P. O. Box 365
Sandia Park, NM 87047

Catherine Meadows
Center for High Assurance Computer Systems
Naval Research Laboratory
Washington, DC 20375

Abstract

Paradoxically, one of the most important – and at the same time, probably one of the least understood – functions performed by information integrity protocols is to transfer trust from where it exists to where it is needed. Initially in any protocol, there are at least two types of trust: trust that designated participants, or groups of participants, will faithfully execute their assigned function in the protocol and trust in the integrity of the transfer mechanism(s) integral to the protocol. Consequently, almost all protocols enforce a set of restrictions as to who may exercise them – either spelled out explicitly or left implicit in the protocol specification. In addition there may be unanticipated or even unacceptable groupings of participants who can also exercise the protocol as a result of actions taken by some of the participants reflecting trusts that exist among them. Formal methods are developed to analyze trust as a fundamental dimension in protocol analysis and proof.

1 Introduction

Since the notion of having to specify trust relationships – central as it is to the analysis of protocols – is still a relatively unfamiliar one, we start by briefly describing a familiar example which clearly illustrates a transfer of trust. The example is a of key distribution protocol for an open link communication network in which there is a universally and unconditionally trusted key generation center (KGC) with whom each subscriber has a trusted (secret and authenticated) private communication link, such as a DES link using the subscribers private key. Since each subscriber trusts the KGC – to protect his key, to authenticate all subscriber requests and responses and most importantly, to faithfully execute the key distribution protocol, there are several protocols available that make it possible for two subscribers who have had no prior exchange of information to end up in possession of a private session key they both trust to keep secret their communication and to vouchsafe the identity of the other party. In a particularly simple example, if A wishes to set up a secure session with B, he sends a cipher to the KGC (encrypted with the private key he shares with the KGC) identifying B, providing required message identifiers, time stamps etc. The KGC responds with a cipher (also encrypted with A's private key) that includes among other items, the session key generated by the

*This paper is a revised and expanded version of a lecture given by the first author at the IEEE Computer Security Foundations Workshop VI, Franconia,NH, June 15-17, 1993 [Sim93]

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1995		**00-00-1995 to 00-00-1995**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
The Role of Trust in Information Integrity Protocols	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Research Laboratory,Center for High Assurance Computer Systems,4555 Overlook Avenue, SW,Washington,DC,20375**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**11**	

KGC and a cipher encrypted with B's private key that identifies A, contains the session key and message identifiers, time stamps etc. as called for by the protocol. A decrypts this cipher to recover the session key and the cipher which only B (and the KGC) can decrypt. A now sends this cipher to B, who decrypts it to recover the session key, verify who he is communicating with etc. The point is, that trust in the KGC and in the integrity of the subscriber's secure communication links with the KGC has been transferred to a trust in the integrity of the communication link established using the session key between two subscribers who had no prior trusted contact. Simple as this example is, it illustrates most of the essential points involved in the transfer of trust in an information integrity protocol.

Trust can also be passed between individuals. For example, one individual A may wish to grant another individual B the right to perform certain actions, on A's behalf. That is, A trusts B to perform these actions. However, A needs to be able to give B the ability to convince others that A trusts B in this capacity. In other words, A needs some means of passing on her trust in B to others. Thus A might give B some ticket signed by A stating that A trusts B. A need for a means of this sort has given rise to a number of *proxy protocols* by means of which A can pass on her trust in B to those who trust A (see [VAB91], for example, for a discussion).

When we decide to pass on trust, we need to determine, not only what the mechanisms for passing on trust should be, but what the consequences of passing on the trust can be. To give a simple example, suppose that an action can only be performed by A and B acting jointly. If B trusts A and allows A to act on his behalf, then the action can be performed by A acting alone. On the other hand, if the action can be performed only by A and B acting jointly, or A and C acting jointly, and B allows C to act on his behalf, nothing has changed.

In this paper we will restrict ourselves to the problem of examining the propagation of trust in access control systems in which an action can only be performed by certain individuals acting in concert. These systems, known as *shared control schemes* [Sim91], can have extremely complex structures. The simplest shared control schemes are unanimous consent schemes, in which an action can only be performed by n designated individuals all acting in concert. Somewhat more complex is the case in which any k members of a group of n individuals can perform the action, but no k-1 can. This case may be enforced by the use of *threshold schemes*, invented by Blakley and Shamir independently [Bla79, Sha79]. Blakley and Shamir showed how to construct a system so that any k individuals out of n can recover a secret, but no k-1 can. This secret can then be used as a token determine whether or not a group is authorized to perform an action. It is also possible to design shared control schemes for giving authority to any set of individuals out of an arbitrary set of sets. As in the case of threshold schemes, these can be realized by shared secret schemes. In [JMS91] Jackson, Martin, and Simmons showed how every such scheme has a perfect geometric realization.

In this paper we look at the problem of determining the effects of introducing added trust in shared control schemes. Such added trust has the property of weakening the scheme, since, as in our example above, new sets will be empowered to perform the controlled action by this trust. We begin by developing a procedure for determining, given two shared control schemes X and Y, which trusts of individuals in other individuals or sets of individuals can produce Y out of X, or, indeed, if this is possible. Next, we look at the structure of the set of all shared control schemes under the "weaker-than" relationship, for a given set of individuals. McLean [McL88] has shown that this set together with the weaker-than relationship is a distributive lattice. We provide an algorithm for constructing the lattice. Finally, we discuss how this lattice can be used in determining the possible effects of passing on trust.

2 Trust as a Mathematical Parameter

The simplest example of conditional trust involves two participants, A and B, each of whom is of questionable trustworthiness when acting alone, but who when compelled to act jointly by an appropriate protocol are considered to be acceptably trustworthy. We will denote this situation by the logical term AB, read to mean A and B acting jointly to exercise whatever function they have in the protocol. We won't address here the critical question of who sets up the protocol, but will simply assume that a protocol has been put in place to enforce a specified trust relation. If AB is the trust relation specified in a protocol, as was the case in our earlier example, and if A trusts B, i.e., if he trusts B enough to give him his proxy, then B acting solely on his own can exercise the protocol. Similarly, if B trusts A, A would be able to act alone, and if both of these trust relations hold, then either A or B acting alone can exercise the protocol. These relationships are illustrated in Figure 1.

Figure 1.

Definition 2.1 We will denote by the symbol , the set of subsets of the participants who are able to exercise a protocol; called either a concurrence scheme (or scheme when the meaning is clear from the context) or an access structure. We use the notation , $= x_1 + ... + x_n$ where the x_i's are the members of , . Each element x of , is in turn denoted as $A_1...A_n$ where the A_j's are the elements of x. Thus, if , is $\{\{A,B\},\{B,C\}\}$, we denote this as , $= AB + BC$. The terms in , are called concurrences. The , are monotone logical functions, meaning that no concurrence in , is a restriction of another concurrence: A and AB cannot both be in the same scheme since AB is a restriction of A.

Using our observations, it is possible to characterize the partially ordered lattice of access structures and to determine the trust relationships that are necessary to move from one element to another in the lattice.

Clearly, trust relations among participants can only weaken the control enforced by a protocol, never strengthen it, i.e., if a specified set of participants can exercise the protocol, trust relations amongst some of them may result in a proper subset of them being able to do so, but the original set will still have the capability: AB weakening to A is possible, A strengthening to AB is impossible. Similarly, trust relationships can make it possible for participants (or sets of participants), not originally empowered to exercise the protocol, becoming able to do so. For example, in the two-person example, A might trust a third person C and share with him his capability. In that case, the original concurrence scheme in Figure 1 would become $AB + BC$ instead of AB. If A also trusted B, this would weaken to B. If B trusted AC this would weaken to $AB + AC + BC$.

We make the assumption that trust is transitive, that is, if X trusts Y, and Y trusts Z, then X trusts Z. This has as a result that all transfers of trust are from individuals to other individuals or groups. For suppose that XY trusts AB. Then, since X trivially trusts XY, we have that X trusts AB.

Using these observations, we can construct the diagram of possible weakenings given by Figure 2.

Figure 2.

Trivial as the example in Figure 2 may seem, it suffices to illustrate how the lattice of concurrence schemes can be used to identify trust relations. Assume that a protocol designer has devised a protocol that requires

B to cooperate with either A or C in order to exercise their function in the protocol, i.e. the control lattice shown in Figure 2, and let $,_1 = AB + BC$ and $,_2 = B + AC$. The subsets B and AC in $,_2$ could only have acquired the capability to exercise the protocol from some subset that had the capability in $,_1$. The notation we use to show this is $AB+BC \rightharpoonup B+AC$, meaning that the concurrence on the left has empowered (entrusted) the concurrence on the right. Suppose that we want to determine how $B + AC$ acquired the capability via trust transfers by individuals. We do this by first expanding the trust transfer in the form

$$((AB \rightharpoonup B) \sqcup (BC \rightharpoonup B)) \sqcap ((AB \rightharpoonup AC) \sqcup (BC \rightharpoonup AC)).$$

where $\sqcup$ stands for logical "or" and $\sqcap$ stands for logical "and".

For each term $x \rightharpoonup y$, we determine what transfers of trust from members of x to factors of y will result in x weakening to y. Consider $AB \rightharpoonup AC$, for example. The choices for A to trust are A, C, and AC. The choice of AC is nonsensical, since it is a strengthening of A. A trusting A is a tautology. If $A \rightharpoonup C$, we have AB weakening to BC, which is not the term we wanted. B's possible trusts that would produce AC out of BC are B trusting A or B trusting AC. The first would turn AB to A, which is weaker than what we want. The second would turn AB to AC even without A trusting C, so A's trust is unnecessary in this case. We now look at B's possible trusts. If $B \rightharpoonup A$, then $AB \rightharpoonup A$ as above. If $B \rightharpoonup C$, then $AB \rightharpoonup AC$, and if $B \rightharpoonup AC$, as above, then $AB \rightharpoonup AC$. $B \rightharpoonup C$ and $B \rightharpoonup AC$. The calculations for $BC \rightharpoonup AC$ are similar, giving us $B \rightharpoonup A$ or $B \rightharpoonup AC$.

The computation of the trusts that result in $AB \rightharpoonup B$ are simpler. The only possible transfer is $A \rightharpoonup B$. Similarly, $BC \rightharpoonup B$ yields $C \rightharpoonup B$.

Combining these expressions, we formally get as the family of trust relations that will weaken $,_1$ to $,_2$:

$$((A \rightharpoonup B) \sqcup (C \rightharpoonup B)) \sqcap ((B \rightharpoonup C) \sqcup ((B \rightharpoonup AC) \sqcap (B \rightharpoonup A)))$$

which when expanded becomes six sets of trust relations, that weaken $,_1$ to the access structures shown on the right:

$$\begin{array}{ll}
(A \rightharpoonup B) \sqcap (B \rightharpoonup C) & B + C \\
(A \rightharpoonup B) \sqcap (B \rightharpoonup AC) & B + AC \\
(A \rightharpoonup B) \sqcap (B \rightharpoonup A) & A + B \\
(C \rightharpoonup B) \sqcap (B \rightharpoonup C) & B + C \\
(C \rightharpoonup B) \sqcap (B \rightharpoonup AC) & B + AC \\
(C \rightharpoonup B) \sqcap (B \rightharpoonup A) & A + B
\end{array}$$

From this we see that there are precisely two sets of trust relations that weaken $,_1$ to $,_2$. The other sets of trust relations that occur in the formal expansion in the example weaken $,_1$ to other elements in the lattice which are themselves dominated by $,_1$. There is no logical inconsistency to this, since the combined effect of multiple trusts can affect (weaken) more terms in a concurrence than just those considered to derive the individual trusts in the first place.

It should be clear from these small examples that the calculation of the set of individual trust relations needed to weaken one concurrence scheme to another is essentially an exercise in applying the distributive laws for the reduction and\or expansion of logical expression and determining the possible trust transfers that will give rise to each term.

3 Dominance of Control

In the last section, we showed how to determine the trust relationships that will produce one concurrence scheme from another. In this section, we study the structure of the "weaker-than" relationship between concurrence schemes. In particular, we show how to construct the lattice of concurrence schemes under this relationship.

Definition 3.1 We say that a concurrence x dominates a concurrence y, written $x \geq y$, or that x is a restriction of y, if y is a subset of x. We say that a concurrence scheme Γ_1, dominates another scheme Γ_2, written $\Gamma_1 \geq \Gamma_2$, if every concurrence in Γ_1 dominates at least one concurrence in Γ_2. We say that Γ_1 covers Γ_2 if there is no Γ_3 such that $\Gamma_1 > \Gamma_3 > \Gamma_2$.

We note that the notion of dominance captures the "weaker-than" relationship in which a concurrence scheme is made weaker by introducing new permissions. One weakens a concurrence scheme by adding new concurrences, and removing concurrences that are supersets of the new concurrences. Thus the old concurrence scheme will dominate the new, weaker, concurrence scheme.

If for example, $\Gamma_1 = AB + AC$ and $\Gamma_2 = A + BC$, then $\Gamma_1 > \Gamma_2$ since AB and AC are restrictions of A, while BC is not a proper restriction of either AB or AC. On the other hand, if $\Gamma_1 = AB + AC$ and $\Gamma_2 = C$, Γ_2 does not dominate Γ_1 since even though AC is a restriction of C, AB is not. Γ_2 does not dominate Γ_1 since C is neither a restriction of AB or AC.

We state and prove the following simple results in a lemma.

Lemma 3.2 Let Γ_1, Γ_2, and Γ_3 be concurrence schemes. Then:

a) If $\Gamma_1 \subset \Gamma_2$, then $\Gamma_1 > \Gamma_2$.

b) If $\Gamma_1 > \Gamma_2$, then no concurrence of Γ_2 strictly dominates a concurrence in Γ_1.

c) If $\Gamma_1 \leq \Gamma_3 \leq \Gamma_2$, then $\Gamma_1 \cap \Gamma_2 \subseteq \Gamma_3$.

Proof: The proof of a) follows from the fact that every element of Γ_1 dominates itself in Γ_2.

To prove b), note that if $x \in \Gamma_2$ strictly dominates y in Γ_1, then by definition of the domination of Γ_1 over Γ_2, there is a z in Γ_2 such that y dominates z. But then x strictly dominates z, which contradicts the monotone property of concurrence scheme.

To prove c), note that if $x \in \Gamma_1 \cap \Gamma_2$ then $x \geq y \in \Gamma_3$ and $y \geq z \in \Gamma_1$. Hence $x \geq z$, where both are in Γ_1. By the monotonicity of Γ_1, we have $x = y = z$. □

It is easy to verify that the dominance relation defines a partial ordering on access structures which can be extended to a lattice with greatest lower bound of Γ_1 and Γ_2 to be the access structure obtained by taking $\Gamma_1 + \Gamma_2$ and removing any element of $\Gamma_1 + \Gamma_2$ that is dominated by any other element of $\Gamma_1 + \Gamma_2$, and least upper bound of Γ_1 and Γ_2 to be the access structure obtained taking $\Gamma = \Gamma_1, \Gamma_2$ and removing any element of Γ that is dominated by any other element of Γ. We note that this results in a lattice similar to the lattice described by McLean in [McL88]. The difference between the lattice described here and McLean's is that, instead of not allowing an access structure Γ to contain two elements such that one dominates

the other, McLean requires that, if x is an element of , then , must also contain all concurrences that are restrictions of x. The partial order defined by McLean is also the reverse of ours. If we reverse the order on McLean's lattices, this allows us to define a natural isomorphism between this lattice and ours by mapping a concurrence scheme , in McLean's lattice to the set of all minimal elements of , , which will define a concurrence scheme in ours. Since the definition of McLean's is one of the standard definitions of the free distributive lattice on n generators [Bir67], we can conclude that the lattice of access structures on n participants is dual-isomorphic to the free distributive lattices on n generators.

Although the lattice of access structures can in principle be computed and exhibited, in practice it is infeasible for even small numbers of participants. The lattice for three participants, L_3 , is shown in Figure 3.

Figure 3.

There is an elegant and simple way, however, to construct both the lattice L_n and the sublattices of L_n consisting of the concurrence schemes reachable from a specific access structure as a consequence of trust relations that may exist between the participants – up to sizes that probably exceed our ability to meaningfully specify trust relations in the first place. Even more important from a design standpoint, given any particular access structure , , this technique will generate and exhibit only the sublattice reachable from , in L_n.

Definition 3.3 Given a lattice L with underlying set S, and a subset $A \subset S$, define the closure of A in S, denoted by A^c, to be the set of elements in S that dominate at least one element of A in L. As is usual in lattice theory, we adopt the convention that $A \subseteq A^c$. The complement of A^c in S is denoted by $S \backslash A^c$. Define two sets, $\lfloor A \rfloor$ and $\lceil A \rceil$, which are most naturally described by the names given these quantities in numerical analysis; the floor and ceiling functions respectively. As will be apparent in a moment, this nomenclature has an intuitive appeal in the setting in which it is used here. $\lfloor A \rfloor$ is the set consisting of the lowest element in all maximal length chains in A^c in L. Similarly, $\lceil A \rceil$ is the set containing the highest element in all the maximal length chains in S$\backslash A^c$ in L.

For example, let L be the lattice defined on the four subsets of the set of two elements under set inclusion, $\emptyset$, $\{a\}$, $\{b\}$, and $\{a, b\}$. Let $A = \{\{a\}\}$. Then $\lfloor A \rfloor = \{\{a\}\}$, and $\lceil A \rceil = \{\{b\}\}$.

The lattice defined on the 2^n subsets of n elements partially ordered by the usual ordering of set inclusion is isomorphic to the binary n-dimensional hypercube H_n. The image of H_n under the floor function is precisely the family of all access structures on n participants, i.e. precisely the set of all possible ways to share capability among n participants. These access structures are in turn partially ordered by the operation of weakening as a result of trust relations as has been described already. The resulting lattice, L_n, is constructively defined by the following theorem:

Theorem 3.4 Let H_n be the lattice of subsets of n elements partially ordered by set inclusion. Let S be the underlying set of H_n. Construct the lattice L_n of access structures on n participants by letting the elements of L_n be sets of elements of H_n. In the lattice, L_n , of access structures on n participants, , $_1$ covers , $_2$ if and only if there exists an x in $\lceil$, $_1\rceil$ such that

$$ll, _2 = (, _1 \setminus S_x) \cup x \quad where\ x \in \lceil, _1\rceil\ and\ S_x = \{y \mid y \in , _1,\ y > x\ \in\ L_n\}.$$

Proof: We will begin by showing that Γ_1 covers $\Gamma_2 = \Gamma_1 \backslash S_x \cup \{x\}$. First we show that $\Gamma_1 > \Gamma_2$. Clearly every element of $\Gamma_1 \backslash S_x$ is a dominates an element of $\Gamma_1 \backslash S_x$. Moreover, every element of S_x dominates x.

Suppose now that there is a Γ_3 such that $\Gamma_1 \geq \Gamma_3 \geq \Gamma_2$. We need to show that either $\Gamma_3 = \Gamma_1$ or $\Gamma_3 = \Gamma_2$. We will first show the conditions under which y can be a member of Γ_3. Suppose that $y \in \Gamma_3$. Then $y \geq z$ where $z \in \Gamma_1 \backslash S_x$, or $y \geq x$. In the first case, y cannot strictly dominate z, by part b) of Lemma 3.2. Thus $y = z$. In the second case, if $y \geq x$, then $y = x$, or $y \in \Gamma_1^c$ by the definition of ceiling. We argue that if $y \in \Gamma_1^c$, then $y \in \Gamma_1$ and hence S_x. For if not, there would be an element z of Γ_1 such that $y > z$, and this would contradict the fact that $\Gamma_1 > \Gamma_3$, by part b) of Lemma 3.2. This shows that Γ_3 is a subset of $\Gamma_1 \cup x$.

Our next step is to show that Γ_3 contains all of $\Gamma_1 \setminus S_x$, and either x or S_x, but not both. In that case, we will have shown that either $\Gamma_3 = \Gamma_1$, or $\Gamma_3 = \Gamma_2$, by the reasoning above. Since both Γ_1 and Γ_2 contain $\Gamma_1 \setminus S_x$, so must Γ_3 by part c) of Lemma 3.2. By part b) of Lemma 3.2, Γ_3 cannot contain both x and any element of S_x. If it contains neither, then $\Gamma_3 \subseteq \Gamma_1$ and hence dominates it by part a) of Lemma 3.2. Thus, Γ_3 must contain either x or an element of S_x. If it contains x, we are done. If it contains a member of S_x, then by part a) of Lemma 3.2, $\Gamma_3 > \Gamma_1$ unless Γ_3 also contains all of S_x.

Next we show that, if Γ_1 covers Γ_2, then $\Gamma_2 = \Gamma_1 \backslash S_x \cup \{x\}$. Since by part a) of Lemma 3.2 Γ_2 is not a subset of Γ_1, there must be an element x of Γ_2 not in Γ_1. This element x dominates no element of Γ_1, so $x \in S \backslash \Gamma_1^c$. If we construct Γ_3 by replacing x with an element $y > x$ in $S \backslash \Gamma_1^c$, then $\Gamma_1 > \Gamma_3 > \Gamma_2$, so x must be maximal, that is $x \in \lceil \Gamma_1 \rceil$. Furthermore, by part b) of Lemma 3.2, Γ_2 contains no element of S_x. As a result, we have that $\Gamma_1 > \Gamma_1 \backslash S_x \cup \{x\} > \Gamma_2$. Hence we have $\Gamma_1 \backslash S_x \cup \{\mathrm{x}\} = \Gamma_2$. □

$\lceil \Gamma \rceil$ is easy to calculate for an access structure Γ. In 1991 Jackson, Martin and Simmons [JMS91] proved that every shared control scheme has a perfect realization by a geometrical scheme. As an essential step in their proof they defined a quantity Γ^* which was computed from the concurrence Γ by interchanging the multiplicative operation and +. Γ^* can be used to compute $\lceil \Gamma \rceil$ as outlined in the following theorem.

Theorem 3.5 Let Γ be a concurrence scheme. Let S be the set of participants. Then $\lceil \Gamma \rceil$ can be computed according to the following steps:

1. Compute $\vec{\Gamma}$ by interchanging the + and x operations.
2. Compute Γ^* by removing each additive term y of $\vec{\Gamma}$ for which there exists a term z of $\vec{\Gamma}$ such that $y > z$.
3. Compute $\overline{\Gamma^*}$ by substituting the set complement of z in S for each additive term z of Γ^*.

Then $\overline{\Gamma^*}$ is $\lceil \Gamma \rceil$.

Example:

$$\begin{aligned} \Gamma &= AB + ACD \\ \vec{\Gamma} &= A + AC + AD + AB + BC + BD \\ \Gamma^* &= A + BC + BD \\ \lceil \Gamma \rceil &= AC + AD + BCD. \end{aligned}$$

Proof:

We first note that computing $\vec{,}$ is equivalent to finding the expression consisting of all terms $y = A_1...A_n$ such that A_i is an element of y_i, where y_i is the i'th term of , . Thus, $\vec{,}$ is the expression containing all terms y such that such that y has at least one element in common with each term of , . Thus, each term z of $\overline{\vec{,}^*}$ has the property that z is missing at least one element of each term of , , so z does not dominate any term of , .

In other words, $\overline{\vec{,}^*}$ is constructed from , as follows:

1. $\vec{,} = \{\text{range}(f) \mid f \text{ is a choice function on } , \}$;
2. $,^*$ is the set of minimal elements of $\vec{,}$, and;
3. $\overline{\vec{,}^*} = \{S \setminus x | x \in ,^*\}$.

We begin by showing that $\overline{\vec{,}^*} \subseteq \lceil , \rceil$. By the argument given above, $\overline{\vec{,}^*} \subseteq S \setminus ,^c$. What remains to show is that each element is maximal.

Suppose that $y \in \overline{\vec{,}^*}$. We need to show that, if A is a member of the alphabet of S not in y, then Ay is in , . We will show that Ay dominates some z in , . Since $y \in \overline{\vec{,}^*}$, $y = S \setminus x$ for some x in $,^*$. Thus $x = \text{range}(f)$ for some choice function f on , . Note that $A \in (S \setminus y = x)$. We claim that there exists z in , with $f(z) = A$ and $z \setminus A$ disjoint from $x \setminus A$. Otherwise, we could construct g mapping some element of $z \setminus A$ to an element of $(z \setminus A) \cap (x \setminus A)$ for every term z with $f(z) = A$, and the same as f elsewhere, and then range(g) would be a subset of $x \setminus A$. This means that x is not minimal, contradicting membership in $,^*$. Hence, choose z in , with $f(z) = A$ and $z \setminus A$ disjoint from $x \setminus A$. But then $z \setminus A$ is a subset of $S \setminus (x \setminus A) = A(S \setminus x) = Ay$, therefore z is a subset of Ay.

We will now show that $\lceil , \rceil \subseteq \overline{\vec{,}^*}$. Suppose that $y \in \lceil , \rceil$. Then, for each term x of , , y fails to dominate x. Thus, there is an element A of x such that y does not contain A. This means that there is a term z of $\overline{\vec{,}^*}$ such that $z \geq y$. But, since y is maximal in S $\setminus ,^c$, we have $z = y$. □

As an example, consider the case $n = 2$, and let $,_1 = a$. Then Then $\lceil ,_1 \rceil = b$, and S_b is empty. Thus the only element covered by $,_1$ is $(,_1 \setminus S_b) + b = a + b$. As another example, let $n = 3$, and let $,_1 = a + bc$. In that case, $\lceil ,_1 \rceil = c + b$. $S_b = bc$, and $S_c = bc$. Then $(,_1 \setminus S_b) + b = a + b$, and $(,_1 \setminus S_c) + c = a + c$.

Given a concurrence , on n participants, to calculate the sublattice L_Γ of concurrences reachable from , in L_n, one first calculates $\lceil , \rceil$ and then calculates all of the concurrences directly dominated by , in L_n using the construction defined in the theorem. The concurrences directly dominated by each of these can then be calculated in turn etc., until the process eventually stops on the concurrence in which each of the n participants is able to exercise the protocol by himself. As we pointed out earlier, the lattice defined by the construction given in the theorem starting with the unanimous consent scheme in which the concurrence of all n of the participants is needed to exercise the protocol is dual-isomorphic to the free distributive lattices with n generators. This is an important observation, since this guarantees that the iterative procedure just described for generating L_Γ will always terminate in at most 2^n steps, since this is the height of the free distributive lattice with n generators (when the empty set is included). Using the observation that L_n is dual-isomorphic to the free distributive lattice with n generators, we can draw on known results in lattice

theory to tabulate the number of ways, N_n, that the capability to exercise a protocol can be shared among n participants.

n	N_n
1	1
2	4
3	18
4	166
5	7,579
6	7,828,352
7	2,414,682,040,996

Table 1

The super-exponential increase in the number of ways that capability can be shared as a function of the number of participants makes clear why only modest numbers are usually considered in practice. Figures 1 and 3 depict L_2 and L_3 respectively. However even L_4 would be difficult to exhibit in a practical sized figure. A useful reduction in the lattice can be achieved by exhibiting equivalence classes (under the group of permutations of element labels) of subsets, instead of the subsets themselves. Figures 4 depicts these lattices for n up to 4, with a representation of each equivalence class indicated for the case $n = 4$. The complete lattic would contain 166 nodes instead of the more manageable 28 equivalence classes shown.

4 The Utility of L_n

Implicitly, when a protocol designer decides that he is willing to accept the risk that a particular concurrence of the participants will all collude to cheat by improperly exercising their capabilities in the protocol, he has also decided that the risk of any subset of them colluding to cheat is unacceptably high. In principle this assumes that he has considered all subsets of the concurrences of participants who are trusted to jointly execute the protocol and has decided that they all represent unacceptable risks. In practice, the decision is made the other way around. The designer decides that the risk of collusion is acceptable for some concurrence, and then gives this group of participants the capability to jointly exercise the protocol.

The utility of the lattice L_n or more specifically by the sublattice L_Γ of L_n generated by a particular concurrence , is that it allows the designer or analyst to precisely evaluate the risk he is accepting. Any set of participants that have the capability to exercise the protocol can share this capability with any other set of persons they trust. The nature of these trust relations can be as complex as the trust relations available to the designer of the protocol in the first place. In a scheme involving n persons, any participant has all of the concurrences possible on n-1 participants available to him to represent his trust of groupings of them. As shown in Table 1, there are 7579 access structures on 5 participants, i.e. 7579 irreducible and distinct ways of entrusting a capability to subsets of 5 participants. Each of these 7579 structures is a possible trust relation by which a participant in a scheme involving 6 participants might conceivably be willing to entrust his share in a shared control scheme to subsets of the other participants. Consequently, given any access

structure, , ,on 6 participants, there are potentially $(7579)^6 \approx 1.9$ x 10^{23} possible sets of trust relations to be considered. Even for such a modest number of participants this is clearly an infeasible number of cases to examine. On the other hand, given an access structure, , , and the sublattice defined by it, L_Γ , it is an easy matter to define the complete family of trust relations that weaken , to any access structure, , $_i$, it dominates in L_Γ. An example of the procedure by which this is done was given earlier.

To be more general, let , $= \Sigma C_i$ and , $_i = \Sigma D_i$. If the set D_i is not an element in , , then one of the subsets in , must have shared their capability with the members of D_i in order for them to have acquired the capability. Any set in , arrows each set in D_i and as we have seen, the write down rules by which a set of participants who possess the capability to exercise a function in a protocol can transfer it to a set who do not are straightforward to see and to mechanize. It is therefore possible to formally construct the entire family of trust relations that could weaken , to , $_i$. It is then necessary to evaluate the concurrence actually implied by each member of this family, since as we've seen some of them may be dominated by , $_i$ itself.

L_n has two primary applications. Since starting from a given concurrence , $_1$ it is only possible to realize as a consequence of trust relationships that may exist among the participants those concurrences dominated by , $_1$, L_n is a primary tool in protocol analysis. If a protocol has been devised to enforce a concurrence , $_1$, then all of the concurrences dominated by , $_1$ in L_n must in a sense also be acceptable risks, since trust relationships between the participants, over whom and which the protocol designer has no control, can realize any of these. Figure 2 is an example of this. We constructed the lattice in that case by first assuming that A trusted C so that the initial concurrence was , $_1 = AB + BC$, but the result is simply the sub lattice in L_3, dominated by , $_1$: see L_3 , in Figure 4.

The second use for L_n is: given an initial concurrence , $_1$ and another concurrence , $_2$ that it dominates, the entire family of trust relations that weaken , $_1$ to realize , $_2$ can be easily calculated. Using this procedure given any initial concurrence , $_1$ and another concurrence, , $_2$, reachable from it, it is easy to enumerate the complete family of sets of trust relationships that could cause this to happen. The protocol designer can then consider each of these sets and decide whether it represents an unacceptable risk or not. From nonprotocol considerations, he may rule out some trust relationships as too unlikely of occurrence to worry about. If the unacceptable concurrences all require sets of trust relations that are judged to be so improbable of occurrence as to be an acceptable risk, then , $_1$ itself represents an acceptable risk. On the other hand, if the trust relations that result in , $_2$ represents an unacceptable risk for the protocol, he may be compelled to choose a different , $_1$ so that , $_2$ becomes either unreachable or dependent on trust relationships considered sufficiently unlikely to occur to provide the desired level of security for the protocol.

The discussion thus far has focused on the question of which groupings of participants can acquire the capability to exercise a function in a protocol as a consequence of trust among some of them. More relevant to the question with which this paper began is the converse question of which trusts can be established as a consequence of those initially in place. We have already shown that it is computationally infeasible to work forward through all possible trust relation to calculate all of the reachable states – even for very modest numbers of participants – however, for the deliberate transfer of trust in a protocol, the protocol specifies the trust relations at the beginning and hence it is only necessary to calculate those that can be reached in consequence. This is a feasible calculation for even large numbers of participants.

5 Conclusion

The notion of trust as a fundamental dimension in protocol analysis or proof has been introduced as well as a methodology that makes it possible to rigorously compute the consequences of trust relations. This methodology makes it possible for the protocol designer to examine in advance all of the possible concurrences of participants who may be able to exercise the protocol as a consequence of trusts that may exist amongst some of them and to decide whether these represent acceptable risks to the integrity of the protocol or not. If not, since the trust relationships are explicitly defined, the methods described here also provide sharp tools for redesigning the protocol to realize acceptable controls. In a sense, instead of the physical principle, "For every action, there is an equal and opposite reaction," in protocol design we have the principle, "For every trust, there is an equal and opposite mistrust."

6 Acknowledgements

We would like to thank the anonymous referees for their helpful comments, especially on the proof for Theorem 3.5.

References

[Bir67] G. Birkhoff. *Lattice Theory*, pages 59–61. American Mathematical Society, Providence, RI, third edition, 1967.

[Bla79] G. R. Blakley. Safeguarding Cryptographic Keys. In *Proc. AFIPS 1979 Natl. Computer Conf.*, pages 313–317, New York, June 1979.

[JMS91] W. A. Jackson, K. Martin, and G. J. Simmons. The Geometry of Shared Secret Schemes. *Bulletin of the Institute of Combinatorics and its Applications (ICA)*, 1(1):71–88, 1991.

[McL88] J. D. McLean. The Algebra of Security. In *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, pages 2–7. IEEE Computer Society Press, April 18-21 1988.

[Sha79] A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.

[Sim91] G. J. Simmons. An Introduction to Shared Secret and/or Shared Control Schemes and Their Application. In G. J. Simmons, editor, *Contemporary Cryptology: The Science of Information Integrity*, chapter 9, pages 441–497. IEEE Press, New York, 1991.

[Sim93] G. J. Simmons. An Introduction to the Mathematics of Trust in Security Protocols. In *Proceedings: Computer Security Foundations Workshop VI*, pages 121–127. IEEE Computer Society Press, June 15-17 1993.

[VAB91] V. Varadharajan, P. Allen, and S. Black. An Analysis of the Proxy Problem in Distributed Systems. In *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 255–275. IEEE Computer Society Press, May 20-22 1991.